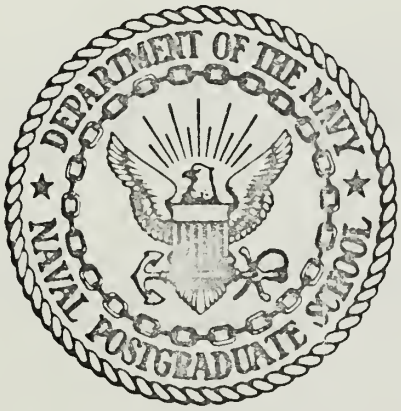
A LINEAR PROGRAMMING METHOD FOR DETECTING NEGATIVE CIRCUITS WITH SPECIAL APPLICATION TO THE ASSIGNMENT PROBLEM

by

Robert Vaughn Dennis

United States
Naval Postgraduate School



THESIS

A LINEAR PROGRAMMING METHOD FOR DETECTING NEGATIVE CIRCUITS WITH SPECIAL APPLICATION TO THE ASSIGNMENT PROBLEM

by

Robert Vaughn Dennis

March 1970

Approved for public release; distribution unlimited.

A Linear Programming Method for Detecting Negative Circuits with Special Application to the Assignment Problem

by

Robert Vaughn Dennis
Lieutenant Colonel, United States Army
B.S., Louisiana State University, 1953

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1970

ABSTRACT

A new method for detecting negative cycles in a graph is proposed. This method is based upon the primal - dual relationships of a linear program formulated from an assignment problem type network. A computer program is developed for this new method to include the complete solution of the assignment problem. Results are given on program efficiency.

TABLE OF CONTENTS

I. OBJECTIVE

A new method has been proposed by H. Greenberg for the detection of negative cycles in an assignment type problem. The objective of this thesis was to develop a computer program for this new method and to measure the efficiency thereof relative to execution time.

II. DESCRIPTION OF THE METHOD

A. INTRODUCTION

Suppose there exists a network with vertices $V = \{1,\ldots,n\}$, directed edges $E = \{(i,j) \in V \times V\}$, a non-negative integral valued function k giving the maximum allowable flow k_{ij} over every edge, and a non-negative cost function a giving the cost a_{ij} associated with a unit of flow over any edge. A flow X of value v is an integral valued function on E satisfying:

$$(1)\quad 0 \leq x_{ij} \leq k_{ij},\ (i,j) \in E.$$

$$(2)\quad \sum_{j=1}^{n} (x_{ij} - x_{ji}) = \begin{cases} v, & i = 1, \\ 0, & i = 2, \ldots, n-1, \\ -v, & i = n. \end{cases}$$

Vertices 1 and n are, respectively, called the flow source and sink.

The minimal cost flow problem is to find, among all flows X of value v, one which minimizes:

$$(3)\quad Q(X) = \sum_{(i,j) \in E} x_{ij} a_{ij}.$$

The assignment problem is to fill n jobs by as many men at least total cost where a_{ij} represents the cost of using man i ($i = 1, \ldots, n$) in job j ($j = n + 1, \ldots, 2n$). Because of this bipartite form the problem can be thought of as a special minimal cost flow problem.

Now suppose a flow has been found satisfying (1) and (2), and for the assignment problem this corresponds to finding a

flow of value $v = n$ and x_{ij} values by, say, the "Least-Cost Rule" given by Dantzig in [1] (p. 307). We then construct an associated network G(X), given by Klein [2], which has the same vertices as the original network and directed edges as follows:

$$(4)\quad E(X) : (i,j) \text{ if } x_{ij} < k_{ij} \text{ and } x_{ji} = 0,$$

$$(j,i) \text{ if } x_{ij} > 0,$$

with revised capacities:

$$(5)\quad k' : k'_{ij} = k_{ij} - x_{ij}, \text{ if } x_{ij} < k_{ij} \text{ and } x_{ji} = 0,$$

$$k'_{ij} = x_{ij}, \text{ if } x_{ij} > 0,$$

and with revised edge costs:

$$(6)\quad a' : a'_{ij} = a_{ij}, \text{ if } x_{ij} < k_{ij} \text{ and } x_{ji} = 0,$$

$$a'_{ji} = -a_{ij}, \text{ if } x_{ij} > 0.$$

These revised edge costs are simply those associated with increasing or cancelling the flow by one unit on these edges: the revised capacities indicate the extent to which this can be done.

Now, use a result proved in Busacker and Saaty ([3], pp. 256-257).

Theorem. X is a minimal cost flow if and only if there is no directed cycle C in G(X) such that the sum of the costs around C's edges are negative. (A directed cycle is a sequence of distinct directed edges of the form $\{(i_o,i_1)\ (i_1,i_2)\cdots(i_p,i_q)\ (i_q,i_o)\}$ involving distinct vertices.)

An immediate consequence of this theorem is that a test of the optimality of the flow X is at hand if G(X) can be checked for the existence of a negative cost directed cycle. Further, if such a cycle is found, an improved flow is obtained by sending a positive unit flow around this cycle. Such a flow alteration obviously leads to a lower total cost and also leaves the flow value v unchanged.

B. THE NEW METHOD

Several methods exist for locating negative cost directed cycles (see Klein [2]). The new method adds to these. The idea behind this method is to formulate a linear program from the network G(X) using the flow balance equations as constraints. A mathematical statement of this set of equations for the assignment problem is:

(7) minimize: $Z = \sum_{(i,j)\in E} x_{ij} a'_{ij}$, where E(X) is determined by the assignment,

subject to:

Flow Balance

$$\sum_{j,(i,j)\in E} x_{ij} - \sum_{j,(j,i)\in E} x_{ji} = 0, \quad i=1,\cdots,n,$$

$$\sum_{i,(j,i)\in E} x_{ij} - \sum_{i,(i,j)\in E} x_{ji} = 0, \quad j=n+1,\cdots,2n,$$

Requires positive flow

$$\sum_{i=1}^{n} \sum_{j=n+1}^{2n} x_{ij} \geq 1,$$

$$x_{ij} \geq 0.$$

Obviously, a feasible solution exists to this program. But, more importantly, if a negative circuit exists then the

solution will be unbounded since the sum of the a'_{ij} around the circuit will be negative and the flow around this circuit will be increased with the solution approaching minus infinity.

The consequence of this is that we have at hand another method for detecting negative circuits by capitalizing on the theorem (see Hadley [4]) that an infeasible dual implies an unbounded primal when the primal has at least one feasible solution. Thus the dual of the above program (7) is solved through Phase I, only, of the Simplex Method to test for feasibility. If feasible, then the assignment is optimal. If the dual is infeasible, then the assignment is not optimal and a change of assignment is made. This process is repeated until a feasible dual and, hence, an optimal assignment is obtained.

The formulation of the primal program is now illustrated with a two by two assignment problem example. The cost matrix for this example is (8):

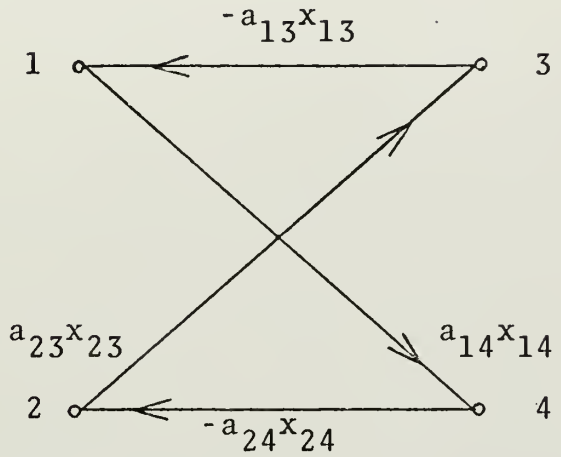
(8)

	3	4
1	a_{13}	a_{14}
2	a_{23}	a_{24}

where i = 1,2 denote the men and j = 3,4 denote the jobs, hereafter referred to as assignee and assigned nodes, respectively. First, an initial assignment must be obtained, say by the "Least-Cost Rule". Suppose this assignment is

(j-i), 3-1, 4-2. Then the graph G(X) representing this problem can be constructed (9).

(9)



The flow balance equations are then (10):

(10)

Node 1: $x_{14} = x_{13}$,

Node 2: $x_{23} = x_{24}$,

Node 3: $x_{13} = x_{23}$,

Node 4: $x_{24} = x_{14}$.

From which is written the linear program (11):

(11) minimize: $Z = -a_{13}x_{13} + a_{14}x_{14} + a_{23}x_{23} - a_{24}x_{24}$,

subject to:

$$\left.\begin{aligned} -x_{13} + x_{14} &= 0, \\ x_{23} - x_{24} &= 0 \\ x_{13} - x_{23} &= 0, \\ -x_{14} + x_{24} &= 0, \end{aligned}\right\} \text{Flow Balance}$$

$$\left.\begin{aligned} x_{13} + x_{14} + x_{23} + x_{24} &\geq 1, \\ x_{ij} &\geq 0. \end{aligned}\right\} \text{Positive Flow}$$

The dual to this program (11) is then solved through Phase I to test for optimality of the assignment.

III. PROGRAMMING THE ALGORITHM

A. THE STANDARD FORMAT

The dual program to (11) is (12):

(12) maximize v_5,

subject to:

$$\begin{array}{rrrrrl} -v_1 & & +v_3 & & \leq & -a_{13}, \\ v_1 & & & -v_4 & \leq & a_{14}, \\ & v_2 & -v_3 & & \leq & a_{23}, \\ & -v_2 & & +v_4 & \leq & -a_{24}, \end{array}$$

which, after conversion to the standard form for solution is (13):

(13) maximize v_5,

subject to:

$$\begin{array}{rrrrrrrrl} v_1 & & -v_3 & & -s_1 & & & = & a_{13}, \\ v_1 & & & -v_4 & & s_2 & & = & a_{14}, \\ & v_2 & -v_3 & & & & s_3 & = & a_{23}, \\ & v_2 & & -v_4 & & & -s_4 & = & a_{24}, \end{array}$$

with slack variables s_1 through s_4. Examining the array of coefficients, a standard format for any size problem can be detected. This format can be illustrated with a three by three problem annotated with general dimensions (14), where n denotes the number of assignee or assigned nodes (in this case n = 3), v_i $(i=1,\cdots,2n)$ the dual variables, $s_j (j=1,\cdots,n^2)$ the slack variables, $-s_j$ the artificial variables added to

(14)

		←n→	←n→	← n^2 →	
	Basic Variables	$v_1 v_2 v_3$	$v_4 v_5 v_6$	$s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9$	Constants
n	s_1	1	-1	1	a_{14}
n	$-s_2$	1	-1	-1	a_{15}
n	s_3	1	-1	1	a_{16}
n^2 n	$-s_4$	1	-1	-1	a_{24}
n^2 n	s_5	1	-1	1	a_{25}
n^2 n	s_6	1	-1	1	a_{26}
n	s_7	1	-1	1	a_{34}
n	s_8	1	-1	1	a_{35}
n	$-s_9$	1	-1	-1	a_{36}
	-w	-1-1-1	1 1 1	0 1 0 1 0 0 0 0 1	$-(a_{15}+a_{24}+a_{36})$

obtain a basis, and W the infeasibility form. Examining the format in detail, it is noted that the first n columns derive from the flow balance constraints on the assignee nodes (numbered from 1 to n); the second n columns derive from the constraints on the assigned nodes (numbered from n+1 to 2n); and the n^2 slack columns reflect the assignment. The slack columns reflect the assignment in that the rows in which the -1's appear indicate the assignment that has been made. If the rows in each group of n rows are numbered identically from n+1 to 2n then the row number in which the -1 appears in the first group of n rows is the number of the node assigned to the first assignee node. Similarly, the remainder of the assignment can be determined. In this example (14), 5 is assigned to 1, 4 is assigned to 2, and 6 is assigned to 3.

Of course, only n of these -1's may appear. Moreover, the row number of a -1 may not be duplicated between groups since this would indicate a duplicate assignment.

It is seen then that the only format change between different problems is in the slack columns reflecting the assignment. This feature was exploited in programming the simplex iterations by reducing the computer storage requirement for this matrix to a single vector of dimensions n^2 by 1. This vector records the assignment and its contents are scanned when format information is needed. It should also be noted that with this format the matrix of coefficients is unimodular. Hence, for every iteration the coefficients will be $\pm$ 1 or 0.

B. ITERATIVE PROCEDURE USED

The revised simplex method was used to perform the Phase I iterations because of the computational advantages. Examining the standard format (14), we see that at least n iterations are required because of the -1's in the first n rows of the infeasibility form equation. To reduce computational time the revised simplex method was modified to carry out these n iterations. The simplex multipliers were not used to find the candidate for the new basic variable. Instead, the first n variables were introduced into the basis sequentially. Subsequently, the standard revised simplex method was used.

At this point it would be useful to illustrate the contents of the matrix used to perform the iterations. Again, this will be illustrated with the previous three by three problem, annotated with general dimensions (15). This matrix is denoted B

(15)

	$\leftarrow n^2 \rightarrow$			
	$s_1\ -s_2\ s_3\ -s_4\ s_5\ s_6\ s_7\ s_8\ -s_9$	Constants	ACOL	VCOL
$\uparrow$	1	a_{14}	0	7
	1	a_{15}	1	-8
	1	a_{16}	0	9
	1	a_{24}	1	-10
n^2	1	a_{25}	0	11
	1	a_{26}	0	12
	1	a_{34}	0	13
	1	a_{35}	0	14
$\downarrow$	1	a_{36}	1	-15
	0 -1 0 -1 0 0 0 0 -1	$-(a_{15}+a_{24}+a_{34})$		

in the computer program. In addition to containing the inverse of the basis and the simplex multipliers, where $-s_j$ denotes the artificial variable added for the corresponding negative slack variable, the matrix contains an assignment column (ACOL) and a variable identification column (VCOL). The assignment column, with binary entries of 0 or 1 only, where 1's record the assignment (in this case the initial assignment is 1-5, 2-4, 3-6), is used as a memory device in performing the iterations and in printing the solution. The variable column records the variables in the basis and is updated after each iteration. The minus indicates an artificial variable. If a problem is infeasible, this column is scanned to determine which artificial variables remain in the basis.

This information is necessary in changing the assignment, which shall be described later.

While the revised simplex method requires less computer storage than the standard simplex method, the storage requirements for this method are, nonetheless, relatively large. For instance a 15 by 15 problem requires approximately 210,000 bytes of storage. (228 X 227 = 51,756 storage locations for the matrix B + 675 storage locations for a computing register and input/output requirements = 52,431 total X 4 bytes per location for single precision = 209,724 bytes.) But, other methods for detecting negative circuits require as much storage.

C. INITIAL ASSIGNMENT

Up to this point an initial assignment has been assumed, but the method for finding that assignment has not been described. The method selected was the "Least-Cost Rule" with steps as follows:

1) The cost matrix is scanned to determine the lowest cost, i.e., find $\underset{(i,j)}{\text{Min}}\, a_{ij}$.

2) The column j and the row i in which this cost appears constitutes the first assignment. That column and row is then ineligible for further assignment.

3) Steps 1) and 2) are repeated for costs remaining in eligible rows and columns until all assignments have been made.

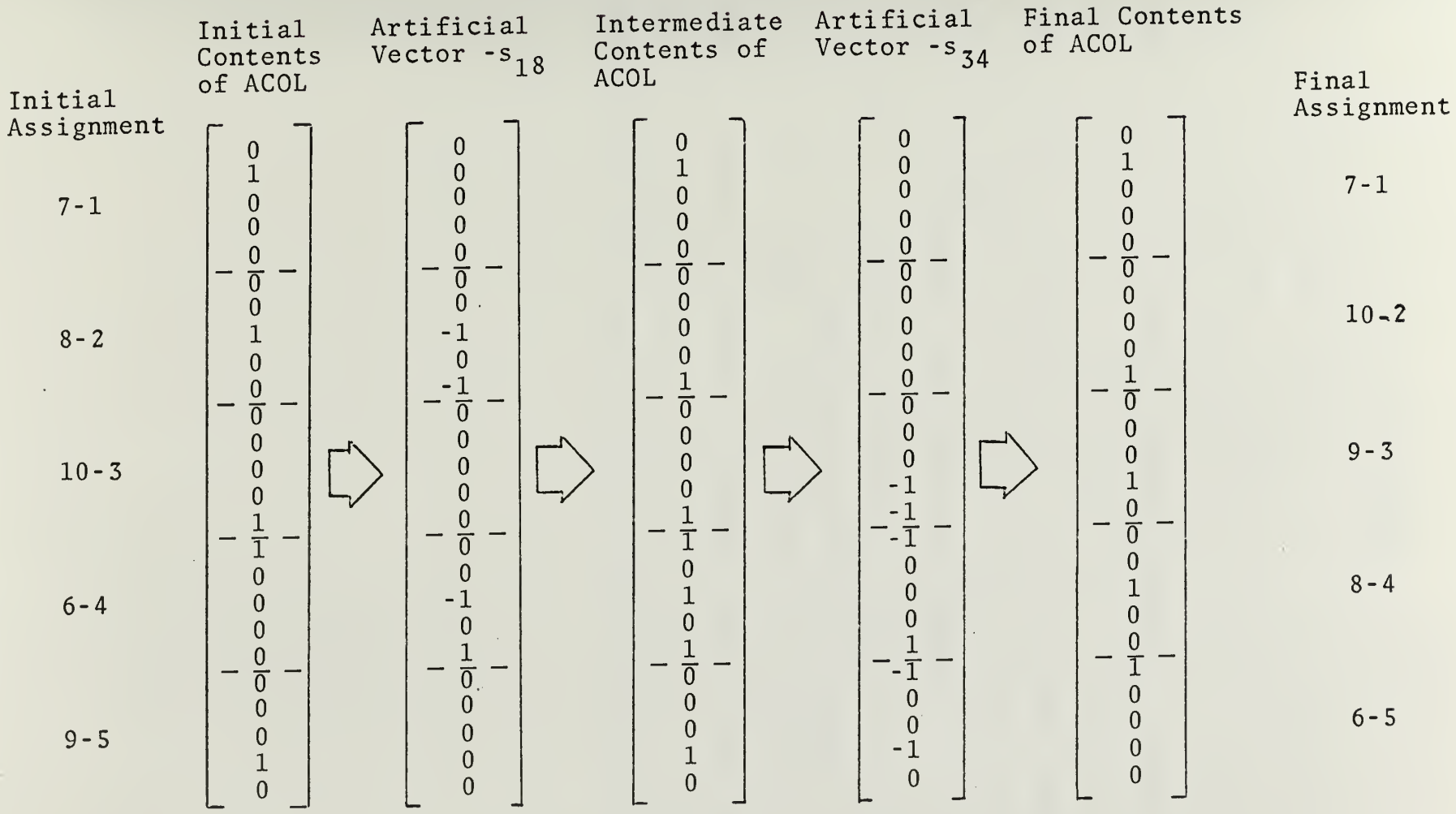
D. CHANGE OF ASSIGNMENT RULE

Summarizing the procedure described thus far - First, an initial assignment is determined. Next the dual problem to (7) is solved through Phase I of the simplex method. If the problem is feasible, then the initial assignment is optimal and the solution is printed. If the problem is not feasible then the initial assignment is not optimal and a change in assignment is required.

The key to devising an assignment change rule is contained in the artificial vectors remaining in the basis because they reflect the infeasibility. In every column of the last n^2 columns in which an entry other than zero appears in the artificial vector, the sign is changed of the $\pm$ 1 in that column of the basic format (14). Thus, the artificial vectors remaining in the basis are identified and constructed from the inverse and the procedure just described is performed for each of these vectors sequentially. The efficacy of this rule is evaluated in the following section. It should be noted that since this procedure does not change the coefficients of any variable in the basis, the inverse for the last tableau remains valid. Therefore, it is not necessary to cycle back to the initial tableau. Rather, after the assignment has been recorded in the memory device (ACOL), the simplex iterations can continue from the last tableau.

A good illustration of application of the assignment rule is problem 5 (see Appendix B), a five by five problem. After nine iterations the program was found to be infeasible and two

artificial vectors remained in the basis ($-s_{18}$ and $-s_{34}$). The vectors shown on the following page trace the change in assignment from initial contents of ACOL to final contents of ACOL. For ease in recognizing the assignments, the vectors have been partitioned by n (n = 5). Recall that the entries of ACOL are binary (0,1) and that an entry other than 0 in the artificial vector "flip-flops" the corresponding ACOL entry.


Initial Assignment
Initial Contents of ACOL
Artificial Vector -s18
Intermediate Contents of ACOL
Artificial Vector -s34
Final Contents of ACOL
Final Assignment
7-1
8-2
10-3
6-4
9-5
7-1
10-2
9-3
8-4
6-5

IV. RESULTS

Twenty problems, ranging in size from four by four to 15 by 15, were tried. The entires comprising the cost matrix were the integers one through nine, selected randomly (see Appendix B). The method yielded an optimal solution in each case. The execution times ranged from 0.7 seconds for a four by four to 56 seconds for a 15 by 15 (see Appendix A). In three instances the initial solution was optimal and, hence, an assignment change was not required. But, in the remaining 17 problems an assignment change was required and in every case the change rule produced the optimal assignment on the first attempt. With respect to the number of iterations required to detect a negative circuit the range was from 7 to 33. And, in all cases the number of iterations required was less than the number equations involved.

APPENDIX A - TIMING RESULTS

Problem Number	Size	Iterations			Execution Time (sec)		
		Detect Negative Circuit	Solve Remainder of Problem	Total	Detect Negative Circuit	Solve Remainder of Problem	Total
1	4X4	7	2	9	0.67	0.03	0.70
2*	4X4	-	-	8	-	-	0.69
3*	4X4	-	-	9	-	-	0.70
4	5X5	8	4	12	0.74	0.10	0.84
5	5X5	9	5	14	0.77	0.13	0.90
6	5X5	10	3	13	0.80	0.07	0.87
7	6X6	11	5	16	0.96	0.23	1.19
8	6X6	9	8	17	0.85	0.39	1.24
9	6X6	8	5	13	0.80	0.24	1.04
10*	7X7	-	-	15	-	-	1.51
11	7X7	11	6	17	1.13	0.49	1.62
12	7X7	13	2	15	1.32	0.12	1.44
13	8X8	15	2	17	1.91	0.18	2.09
14	8X8	14	4	18	1.76	0.49	2.25
15	8X8	15	5	20	1.92	0.63	2.55
16	9X9	13	4	17	1.88	0.74	2.62
17	9X9	17	4	21	2.88	0.74	3.62
18	10X10	21	3	24	5.12	0.73	5.90
19	10X10	19	3	22	4.33	0.75	5.08
20	15X15	33	14	47	34.00	21.97	55.97

*Initial problem was feasible; i.e., initial assignment was optimal

APPENDIX B

PROBLEMS AND SOLUTIONS

1. Problem 1

	5	6	7	8
1	9	4	9	2
2	4	1	5	9
3	4	9	6	1
4	2	9	7	5

Initial Assignment	Optimal Assignment
1-7	1-8
2-6	2-6
3-8	3-7
4-5	4-5

2. Problem 2

	5	6	7	8
1	8	7	3	4
2	5	5	7	4
3	8	8	2	1
4	9	5	9	4

Initial Assignment	Optimal Assignment
1-7	1-7
2-5	2-5
3-8	3-8
4-6	4-6

3. Problem 3

	5	6	7	8
1	5	8	4	1
2	6	8	9	2
3	4	2	9	1
4	6	5	5	3

Initial Assignment	Optimal Assignment
1-8	1-8
2-5	2-5
3-6	3-6
4-7	4-7

4. Problem 4

	6	7	8	9	10
1	5	2	4	7	1
2	3	5	9	9	4
3	2	5	6	1	4
4	3	9	9	3	9
5	7	9	1	6	3

Initial Assignment	Optimal Assignment
1-10	1-7
2-6	2-10
3-9	3-9
4-7	4-6
5-8	5-8

5. Problem 5

	6	7	8	9	10
1	9	2	9	9	6
2	7	8	4	8	2
3	7	4	8	2	1
4	4	6	6	9	5
5	4	5	8	9	8

Initial Assignment	Optimal Assignment
1-7	1-7
2-8	2-10
3-10	3-9
4-6	4-8
5-9	5-6

6. Problem 6

	6	7	8	9	10
1	9	9	5	1	5
2	4	1	3	5	6
3	7	9	7	8	9
4	4	5	6	3	5
5	3	4	2	1	9

Initial Assignment	Optimal Assignment
1-9	1-9
2-7	2-7
3-10	3-6
4-6	4-10
5-8	5-8

7. Problem 7

	7	8	9	10	11	12
1	3	9	2	1	1	9
2	7	1	8	5	2	9
3	5	4	2	5	2	8
4	7	9	9	4	4	8
5	9	1	6	5	3	2
6	6	9	2	3	7	6

Initial Assignment	Optimal Assignment
1-10	1-7
2-8	2-8
3-9	3-11
4-11	4-10
5-12	5-12
6-7	6-9

8. Problem 8

	7	8	9	10	11	12
1	2	8	6	6	5	8
2	5	4	2	1	6	7
3	2	5	5	2	4	5
4	6	1	2	2	8	5
5	6	8	5	8	4	3
6	5	6	6	1	5	4

Initial Assignment	Optimal Assignment
1-7	1-7
2-10	2-9
3-11	3-11
4-8	4-8
5-12	5-12
6-9	6-10

9. Problem 9

	7	8	9	10	11	12
1	1	4	7	3	8	8
2	4	2	7	3	4	2
3	6	4	3	1	2	2
4	9	8	9	3	8	4
5	5	1	5	6	3	3
6	8	6	1	4	8	4

Initial Assignment	Optimal Assignment
1-7	1-7
2-12	2-12
3-10	3-11
4-11	4-10
5-8	5-8
6-9	6-9

10. Problem 10

	8	9	10	11	12	13	14
1	1	5	8	4	3	5	2
2	7	6	7	3	7	9	5
3	5	8	5	1	8	5	6
4	6	7	8	9	7	8	9
5	3	2	2	8	1	4	2
6	8	4	3	8	4	7	7
7	5	7	8	9	9	6	1

Initial Assignment	Optimal Assignment
1-8	1-8
2-9	2-9
3-11	3-11
4-13	4-13
5-12	5-12
6-10	6-10
7-14	7-14

11. Problem 11

	8	9	10	11	12	13	14	Initial Assignment	Optimal Assignment
1	6	9	9	5	3	3	2	1-14	1-14
2	7	6	3	2	7	7	5	2-11	2-10
3	2	1	6	8	8	9	9	3-9	3-9
4	1	1	3	5	4	5	2	4-8	4-8
5	9	3	6	9	4	1	4	5-13	5-13
6	9	3	3	2	2	7	9	6-12	6-11
7	3	2	9	9	3	5	7	7-10	7-12

12. Problem 12

	8	9	10	11	12	13	14	Initial Assignment	Optimal Assignment
1	7	8	9	3	6	7	1	1-14	1-14
2	8	3	7	3	2	3	7	2-12	2-9
3	4	7	6	5	2	7	6	3-10	3-12
4	3	7	8	3	9	7	4	4-11	4-11
5	6	6	8	3	5	1	5	5-13	5-13
6	2	1	3	1	9	9	8	6-9	6-10
7	2	8	9	3	4	7	8	7-8	7-8

13. Problem 13

	9	10	11	12	13	14	15	16	Initial Assignment	Optimal Assignment
1	5	2	1	2	7	9	2	5	1-11	1-11
2	1	6	8	3	3	3	2	7	2-9	2-9
3	4	4	4	2	9	8	3	7	3-12	3-12
4	7	7	3	8	2	9	6	2	4-16	4-16
5	8	9	8	7	3	3	5	5	5-10	5-13
6	3	3	8	4	1	8	9	8	6-13	6-10
7	5	6	1	5	7	9	1	4	7-15	7-15
8	3	3	4	5	7	1	5	1	8-14	8-14

14. Problem 14

	9	10	11	12	13	14	15	16
1	5	9	9	5	5	5	4	9
2	2	8	5	8	3	4	6	4
3	9	7	2	9	9	7	3	7
4	7	2	4	4	5	9	8	6
5	4	4	1	3	9	3	1	4
6	6	8	4	9	9	1	7	8
7	2	8	4	1	8	9	2	2
8	7	6	4	1	1	6	1	6

Initial Assignment	Optimal Assignment
1-16	1-12
2-9	2-9
3-15	3-11
4-10	4-10
5-11	5-15
6-14	6-14
7-12	7-16
8-13	8-13

15. Problem 15

	9	10	11	12	13	14	15	16
1	8	5	6	6	4	7	6	8
2	5	3	8	5	2	2	8	7
3	4	1	5	4	1	2	9	7
4	1	3	3	9	2	3	2	7
5	9	7	1	4	4	2	7	1
6	5	9	2	7	6	5	8	5
7	3	5	5	1	9	5	7	1
8	5	8	5	1	8	9	7	1

Initial Assignment	Optimal Assignment
1-15	1-15
2-13	2-13
3-10	3-10
4-9	4-9
5-11	5-14
6-14	6-11
7-12	7-16
8-16	8-12

16. Problem 16

	10	11	12	13	14	15	16	17	18
1	9	5	7	4	9	1	9	9	1
2	6	7	1	2	4	9	2	7	9
3	7	5	8	3	1	9	7	4	7
4	9	8	3	5	4	2	3	7	5
5	1	3	7	7	8	2	4	5	3
6	7	2	1	8	3	5	1	3	1
7	2	2	4	9	1	7	3	6	6
8	6	6	4	2	6	2	7	1	2
9	5	7	4	1	5	9	6	3	5

Initial Assignment	Optimal Assignment
1-15	1-18
2-12	2-12
3-14	3-14
4-18	4-15
5-10	5-10
6-16	6-16
7-11	7-11
8-17	8-17
9-13	9-13

17. Problem 17

	10	11	12	13	14	15	16	17	18	Initial Assignment	Optimal Assignment
1	9	2	2	2	5	1	7	5	4	1-15	1-12
2	4	5	7	2	5	3	1	2	6	2-16	2-16
3	1	5	1	1	8	8	2	7	6	3-10	3-10
4	5	9	8	3	2	1	1	3	1	4-18	4-15
5	9	6	7	9	4	7	2	3	3	5-12	5-18
6	2	7	5	2	8	4	7	1	4	6-17	6-17
7	4	3	3	4	2	4	5	3	2	7-14	7-14
8	4	6	7	1	6	1	3	6	8	8-13	8-13
9	7	4	7	9	6	8	9	8	4	9-11	9-11

18. Problem 18

	11	12	13	14	15	16	17	18	19	20	Initial Assignment	Optimal Assignment
1	3	5	9	8	8	6	5	8	4	5	1-18	1-11
2	1	9	1	8	3	3	1	8	8	3	2-11	2-17
3	7	7	8	5	9	1	4	3	4	4	3-16	3-16
4	6	8	3	3	5	7	7	8	4	7	4-15	4-15
5	3	3	5	4	5	7	2	4	4	4	5-17	5-18
6	3	2	1	4	3	3	8	8	6	6	6-13	6-13
7	5	8	5	2	4	5	6	7	7	6	7-14	7-14
8	1	4	9	9	9	1	5	7	2	2	8-20	8-20
9	1	5	1	9	6	2	9	9	1	2	9-19	9-19
10	5	1	2	4	8	5	5	3	2	9	10-12	10-12

19. Problem 19

	11	12	13	14	15	16	17	18	19	20	Initial Assignment	Optimal Assignment
1	7	8	6	2	5	2	2	1	7	9	1-18	1-18
2	9	8	1	3	4	3	2	3	3	3	2-13	2-13
3	5	6	1	4	6	1	1	3	2	2	3-16	3-17
4	8	5	6	8	6	4	2	4	6	1	4-20	4-20
5	1	9	6	6	4	9	5	6	4	8	5-11	5-11
6	1	4	4	4	7	9	8	9	1	8	6-19	6-19
7	6	1	4	3	2	4	7	3	1	1	7-12	7-12
8	3	3	9	8	5	2	6	4	2	4	8-15	8-16
9	6	1	4	1	7	1	6	8	4	1	9-14	9-14
10	2	9	8	6	4	7	3	9	7	7	10-17	10-15

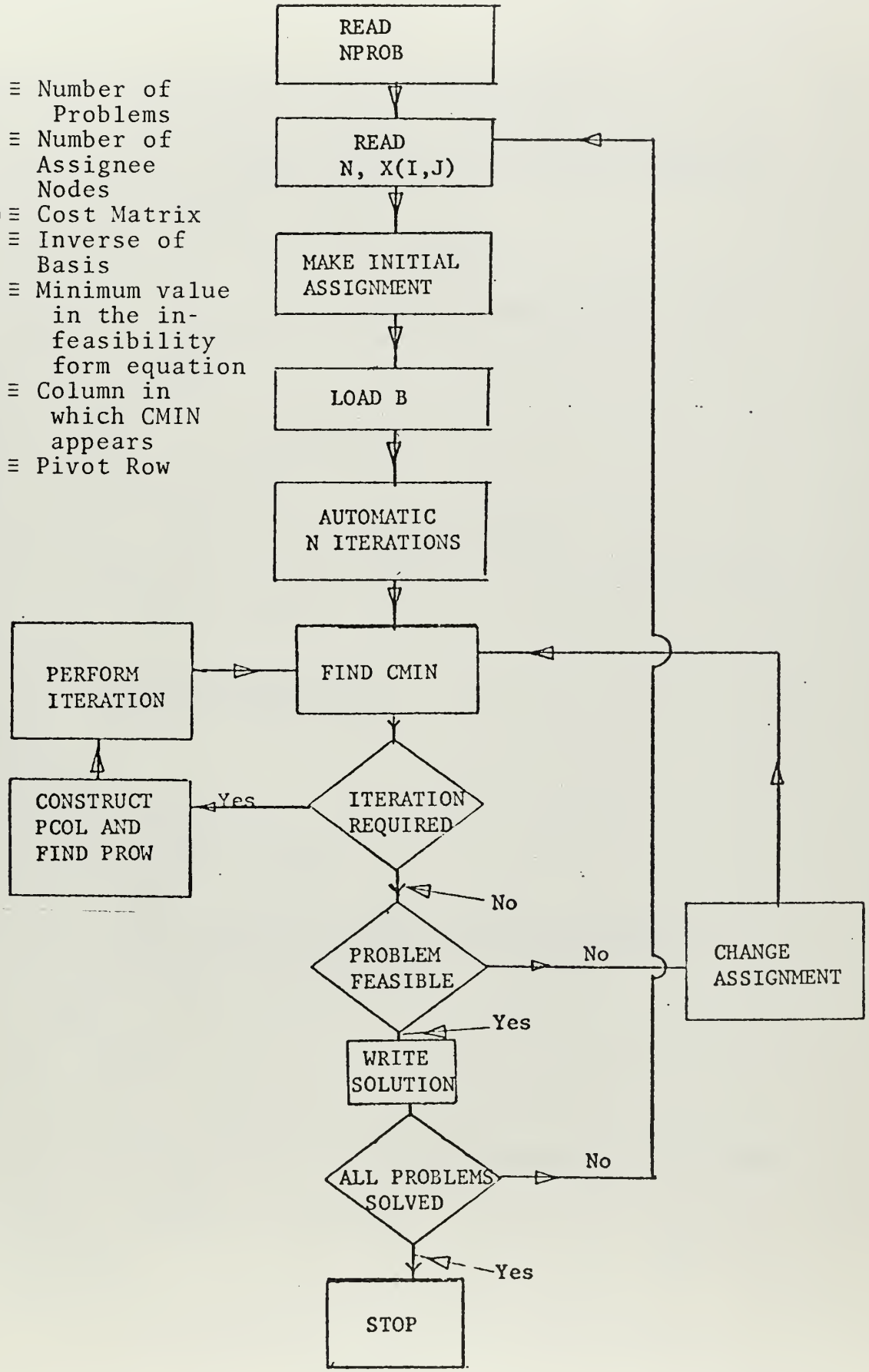
20. Problem 20

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Initial Assignment	Optimal Assignment
1	7	5	7	4	6	5	6	1	7	8	3	5	9	9	1	1-23	1-30
2	2	3	8	8	3	7	4	5	8	9	1	9	5	2	6	2-26	2-29
3	1	6	5	7	9	1	6	3	7	1	5	9	5	3	5	3-16	3-25
4	6	1	1	8	2	4	7	4	9	6	3	3	1	3	7	4-17	4-18
5	8	4	4	1	2	2	3	1	6	3	7	3	4	6	3	5-19	5-19
6	1	2	7	6	3	5	5	5	5	7	8	6	3	6	9	6-29	6-16
7	3	5	6	7	1	6	9	4	7	4	9	4	6	2	9	7-20	7-20
8	5	3	9	4	7	2	5	5	5	3	4	5	7	8	9	8-21	8-21
9	4	7	8	8	4	6	5	1	2	4	6	3	6	8	3	9-25	9-24
10	5	9	5	2	3	6	9	1	8	8	3	3	7	7	7	10-18	10-23
11	1	8	3	9	1	6	8	2	9	4	3	3	1	7	4	11-28	11-28
12	9	5	2	8	7	4	2	8	9	6	2	1	3	7	8	12-27	12-27
13	5	3	4	8	2	3	6	3	3	8	9	3	5	5	2	13-30	13-17
14	8	5	5	6	3	5	2	5	1	2	6	7	8	8	7	14-24	14-22
15	2	3	7	9	5	7	2	1	8	4	1	8	6	8	9	15-22	15-26

APPENDIX C

FLOW DIAGRAM

NPROB ≡ Number of Problems
N ≡ Number of Assignee Nodes
X(I,J) ≡ Cost Matrix
B ≡ Inverse of Basis
CMIN ≡ Minimum value in the in-feasibility form equation
PCOL ≡ Column in which CMIN appears
PROW ≡ Pivot Row



```
      INTEGER X(50,50),AROW(230),L,ACOL,VCOL,K,N,
     1MIN,PROW,CBAR,CMIN,PCOL(230),B(230,230),D
C THE NEXT 33 LINES READ IN THE DATA,ZERO THE INVERSE AND
C DETERMINE THE INITIAL ASSIGNMENT. THE DATA ARE:
C NPROB=NUMBER OF PROBLEMS, N=NUMBER OF ASSIGNEE NODES,
C X(I,J)=COST MATRIX. NOTE THIS PROGRAM IS DIMENSIONED TO
C HANDLE UP TO A 15 X 15 PROBLEM.
      READ (5,10) NPROB
C  SET TIMER TO MEASURE EXECUTION TIMES.
   85 CALL SETIME
      DO 91 I = 1,50
   91 A(I) = 0
C  A(I) STORES THE ASSIGNMENT FOR PRINTING PURPOSES AND TO
C  LOAD ACOL.
      DO 92 I = 1,230
      DO 92 J = 1,230
   92 B(I,J) = 0
      READ (5,10) N
      DO 1 J= 1,N
    1 READ (5,20) (X(I,J),I=1,N)
      WRITE(6,30)
      CALL WRTMTX (X,N,N)
C  WRITES INPUT DATA.
      ICOUNT = 0
C  ICOUNT COUNTS NUMBER OF ASSIGNMENTS MADE.
    2 MIN = 9999
      DO 4 J = 1,N
      DO 5 K = 1,N
      IF(J.EQ.A(K)-N) GO TO 4
    5 CONTINUE
      DO 3 I = 1,N
      IF(A(I).GT.0) GO TO 3
      IF(X(I,J).GE.MIN) GO TO 3
      MIN = X(I,J)
      IROW = I
      JCOL = J
    3 CONTINUE
    4 CONTINUE
      A(IROW) = JCOL + N
      ICOUNT = ICOUNT + 1
      IF(ICOUNT.LT.N) GO TO 2
      WRITE (6,40)
      DO 6 I = 1,N
    6 WRITE (6,50) I,A(I)
C THE NEXT 44 LINES LOAD THE INVERSE OF THE BASIS (B).
C COMMENT IS MADE WHEN EACH ELEMENT OF B IS LOADED.
      ITER = N-1
C  ITER COUNTS NUMBER OF ITERATIONS.
      D = N*N
      L = N*N+1
      ACOL = N*N+2
      VCOL = N*N+3
      K = 1
C  LOAD DIAGONAL.
      DO 8 I = 1,D
    8 B(I,I) = 1
C  LOAD LAST ROW EXCEPT THE CONSTANT (SIMPLEX MULTIPLIERS).
      DO 11 J = 1,N
      DO 11 I = 1,N
      IF(A(J) - N.EQ. I) GO TO 9
      B(L,K) = 0
      GO TO 11
    9 B(L,K) = -1
   11 K = K + 1
```

```
C  LOAD COLUMN OF CONSTANTS EXCEPT LAST ROW.
       K = 1
       J = 1
       DO 12 I = 1,D
       B(I,L) = X(K,J)
       J = J+1
       IF(J.LE.N) GO TO 12
       K = K+1
       J = 1
  12   CONTINUE
C  LOAD ASSIGNMENT COLUMN (ACOL).
       DO 14 K = 1,N
       J = (K*N) -N
       DO 13 I = 1,N
       IF(A(K) - N.NE.I) GO TO 13
       B(J+I,ACOL) = 1
       GO TO 14
  13   CONTINUE
  14   CONTINUE
C  LOAD CONSTANT FOR LAST ROW.
       DO 16 I = 1,D
       IF(B(I,ACOL).EQ.1) GO TO 15
       B(I,VCOL) = 2*N +I
       GO TO 16
  15   B(I,VCOL) = (2*N + I) *(-1)
  16   CONTINUE
       SUM = 0
       DO 18 I = 1,D
       IF(B(I,ACOL).EQ.1) GO TO 17
       GO TO 18
  17   SUM = SUM + B(I,L)
  18   CONTINUE
       B(L,L) = SUM * (-1)
C THE NEXT 23 LINES MAKE THE FIRST N SIMPLEX ITERATIONS.
C THE FIRST 9 LINES FIND THE PIVOT ROW (PROW) BY FINDING
C THE MINIMUM CONSTANT VALUE IN EACH GROUP OF N.
       M = 1
       K = 1
       KSTAR = K + (N-1)
  19   MIN = 9999
       DO 21 I = K,KSTAR
       IF(B(I,L).GE.MIN) GO TO 21
       MIN = B(I,L)
       PROW = I
  21   CONTINUE
C  UPDATE VCOL.
       B(PROW,VCOL) = M
C  PERFORM ITERATION.
       DO 81 J = 1,L
       DO 23 I = K,KSTAR
       IF(I.EQ.PROW) GO TO 22
       B(I,J) = B(I,J) - B(PROW,J)
       GO TO 23
  22   B(I,J) = B(I,J)
  23   CONTINUE
       B(L,J) = B(L,J) + B(PROW,J)
  81   CONTINUE
C  UPDATE INDEXES.
       M = M+1
       K = K+N
       KSTAR = K + (N-1)
       IF(K.LT.N*N) GO TO 19
C THE NEXT 24 LINES FIND THE MINIMUN VALUE IN THE
C INFEASIBILTY FORM EQUATION (CMIN).  JCONT IS A COLUMN
C COUNTER.  JSTAR IS AN INDEX.
  49   CMIN = 1
       JCONT = 1
       JSTAR = D + 2*N
       DO 29 J = 1,JSTAR
       CRAR = 0
C  NEXT 4 LINES DETERMINE THE ARITHMETIC TO BE DONE
C  DEPENDING UPON LOCATION IN STANDARD FORMAT.
```

```
      IF(J.LE.N) GO TO 24
      IF(J.LE.2*N) GO TO 25
      IF(B(JCONT,ACOL).EQ.1) GO TO 26
      CBAR = CBAR + B(L,JCONT)
      GO TO 27
   24 K = (J-1)*N
      DO 51 I = 1,N
   51 CBAR = CBAR + B(L,I+K)
      GO TO 28
   25 K = J - (N+1)
      DO 53 I = 1,D,N
   53 CBAR = CBAR + B(L,I+K)*(-1)
      GO TO 28
   26 CBAR = CBAR +B(L,JCONT) *(-1)
   27 JCONT = JCONT + 1
   28 IF(CBAR.GE.CMIN) GO TO 29
      CMIN = CBAR
      JCOL = J
   29 CONTINUE
C  THE NEXT 21 LINES TEST TO DETERMINE IF AN ITERATION IS
C  REQUIRED OR IF A CHANGE IN ASSIGNMENT IS REQUIRED. IF NOT,
C  THE SOLUTION IS WRITTEN AND THE PROBLEM COUNTER IS TESTED.
      ITER = ITER +1
      IF(CMIN.LT.0) GO TO 31
      IF(B(L,L).NE.0) GO TO 43
      WRITE(6,60) ITER
      K = 1
      DO 86 I = 1,D,N
      DO 87 J = 1,N
      IF (B(I+J-1,ACOL).EQ.0) GO TO 87
      A(K) = N+J
      K = K + 1
   87 CONTINUE
   86 CONTINUE
      DO 88 I = 1,N
   88 WRITE (6,50) I,A(I)
C   MEASURE TIME FROM DETECTION OF NEGATIVE CIRCUIT UNTIL
C   SOLUTION OF PROBLEM.
      CALL GETIME (IET)
      EL = IET * .000026
      WRITE (6,120) NPROB
      WRITE (6,130) EL
      NPROB = NPROB - 1
      IF (NPROB.GT.0) GO TO 85
      STOP
C  THE NEXT 29 LINES CONSTRUCT THE COLUMN IN WHICH CMIN
C  APPEARS (PCOL) AND FINDS THE PIVOT ROW (PROW).
   31 IF(JCOL.LE.N) GO TO 34
      IF(JCOL.LE.2*N) GO TO 35
      K = JCOL - 2*N
      IF(B(K,ACOL).EQ.1) GO TO 32
      DO 37 I = 1,D
   37 PCOL(I) = B(I,K)
      GO TO 36
   32 DO 33 I = 1,D
   33 PCOL(I) = B(I,K)*(-1)
      GO TO 36
   34 K = (JCOL-1) *N
      DO 38 I = 1,D
      PCOL(I) = 0
      DO 38 J = 1,N
   38 PCOL(I) = PCOL(I) + B(I,K+J)
      GO TO 36
   35 K = JCOL -(N+1)
      DO 39 I = 1,D
      PCOL(I) = 0
      DO 39 J = 1,D,N
   39 PCOL(I) = PCOL(I) - B(I,K+J)
   36 MIN = 9999
      DO 41 I = 1,D
      IF(PCOL(I).EQ.1) GO TO 42
      GO TO 41
```

```
   42  IF(B(I,L).GE.MIN) GO TO 41
       MIN = B(I,L)
       PROW = I
C THE NEXT 17 LINES PERFORM THE SIMPLEX ITERATION.
   41  CONTINUE
       B(PROW,VCOL) = JCOL
       DO 76 J = 1,L
       DO 75 I = 1,D
       IF(I.EQ.PROW) GO TO 72
       IF(PCOL(I).EQ.-1) GO TO 73
       IF(PCOL(I).EQ.0) GO TO 72
       IF(PCOL(I).EQ.1) GO TO 74
       GO TO 75
   72  B(I,J) = B(I,J)
       GO TO 75
   73  B(I,J) = B(I,J) + B(PROW,J)
       GO TO 75
   74  B(I,J) = B(I,J) - B(PROW,J)
   75  CONTINUE
       B(L,J) = B(L,J) - (CMIN*B(PROW,J))
   76  CONTINUE
       GO TO 49
C THE NEXT 25 LINES MAKES THE CHANGE IN ASSIGMENT.
C GETIME MEASURES TIME TO DETECT NEGATIVE CIRCUIT.
   43  CALL GETIME (IET)
       EL = IET * .000026
       WRITE (6,110) EL
C  RESET TIMER.
       CALL SETIME
       DO 48 I = 1,D
       IF(B(I,VCOL).GE.0) GO TO 48
       DO 47 J = 1,D
       IF(B(J,ACOL).EQ.1) GO TO 44
       AROW(J) = B(I,J)
C  AROW (J) STORES CONTENTS OF LAST N*N COLUMNS OF
C  ARTIFICIAL VECTOR.
       GO TO 45
   44  AROW(J) = B(I,J)*(-1)
   45  IF(AROW(J).NE.0)  GO TO 46
       GO TO 47
   46  IF(B(J,ACOL).EQ.1) GO TO 83
       IF(B(J,ACOL).EQ.0) GO TO 84
       GO TO 47
   83  B(J,ACOL) = 0
       GO TO 47
   84  B(J,ACOL) = 1
   47  CONTINUE
       WRITE(6,80) B(I,VCOL)
       WRITE (6,90) (AROW(K), K = 1,D)
   48  CONTINUE
       WRITE(6,70) ITER
       GO TO 49
   10  FORMAT(I10)
   20  FORMAT (10I8)
   30  FORMAT (1H1,10X,'INPUT DATA')
   40  FORMAT (////)
   50  FORMAT (1H ,I4,T6,'-',T7,I4)
   60  FORMAT (1H0,5X,'PROBLEM FEASIBLE AFTER ITERATION-',
      1I4,5X,'OPTIMUM ASSIGNMENT IS:')
   70  FORMAT (1H0,5X,'PROGRAM INFEASIBLE AFTER
      1ITERATION-',I4)
   80  FORMAT (1H0,5X,'ARTIFICIAL VECTOR FOR -',I4)
   90  FORMAT (1H0,5X,40I3)
  100  FORMAT (1H0,5X,'COST IS:',I4)
  110  FORMAT (1H0,5X,'TIME TO DETECT NEGATIVE CIRCUIT
      1WAS',F16.6,'SEC')
  120  FORMAT (1H0,5X,'TIME TO SOLVE PROBLEM-',I4,5X,
      1'FOLLOWS')
  130  FORMAT (1H0,5X,F16.6,'SEC')
       END
```

```
      SUBROUTINE WRTMTX (K,IROW,JCOL)
      DIMENSION K(50,50)
      WRITE (6,30) (I, I = 1,JCOL)
30    FORMAT (// 1H ,4X,'I/J=',50I3)
      WRITE (6,31)
31    FORMAT (1H )
32    FORMAT (1H ,3X,I2,3X,50I3)
      DO 33 I = 1, IROW
33    WRITE (6,32) I, (K(I,J), J = 1,JCOL)
      RETURN
```

REFERENCES

1. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, N. J., 1963.

2. Klein, J., A Primal Method for Minimal Cost Flows with Applications to the Assignment and Transportation Problems, Management Science, v. 14, No. 3, November 1967.

3. Busacker, R. G. and Saaty, T. L., Finite Graphs and Networks, McGraw-Hill, New York, N. Y., 1965.

4. Hadley, G., Linear Programming, Addison-Wesley, Reading, Mass., 1962.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Professor Harold Greenberg, Code 55Gd Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
4.	LTCOL Robert V. Dennis 400 Grove Delmar, Delaware 19940	1
5.	Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	Unclassified
	2b. GROUP

3 REPORT TITLE

A Linear Programming Method for Detecting Negative Circuits with Special Application to the Assignment Problem

4 DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; March 1970

5. AUTHOR(S) (First name, middle initial, last name)

Robert Vaughn Dennis

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1970	37	4

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for Public Release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

A new method for detecting negative cycles in a graph is proposed. This method is based upon the primal - dual relationships of a linear program formulated from an assignment problem type network. A computer program is developed for this new method to include the complete solution of the assignment problem. Results are given on program efficiency.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Linear Programming Method						
Negative Circuits						
Assignment Problem						

Thesis
D359
c.1

125797

Dennis
A linear programming method for detecting negative circuits with special application to the assignment problem.